NOTICE: When government or other drawings, specifications or other data are used for any purpose other than in connection with a definitely related government procurement operation, the U. S. Government thereby incurs no responsibility, nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.

NRL Report 5784

ENCORES FOR ACT I SYSTEM FOR LGP-30 DIGITAL COMPUTER COMPILER

Alan H. Rich

Propagation Branch
Sound Division

June 1, 1962

U. S. NAVAL RESEARCH LABORATORY
Washington, D.C.

CONTENTS

ABSTRACT

The Algebraic Compiler and Translator I System for the LGP-30 digital computer has been modified and expanded for greater versatility. A location symbols table is generated at a new location, providing space for one of two decimal program and references printout routine. A 63-symbol operations table includes non-input typewriter operation symbols, breakpoint symbols, a compatible hex or alphanumeric output symbol, a binarization of address symbol, a flexible program input symbol, an integer-trim symbol, and a trace-one-intermediate-result symbol. Two alternate subroutines, located in unused space at the time of computing, utilize program input routine.

PROBLEM STATUS

This is an interim report on the problem; work is continuing.

AUTHORIZATION

NRL Problem S01-01
Project RR 004-07-41-5250

Manuscript submitted April 6, 1962.

ENCORES FOR ACT I SYSTEM FOR LGP-30 DIGITAL COMPUTER COMPILER

INTRODUCTION

Modifying and supplementing subroutines have been written for the LGP-30 digital computer compiler system, Act I (1,2). Called "Encores," the subroutines include compiler modifications which cause the symbol table to be stored and punched out in 6000-6163 after the object program. This modification provides space for either "Cast of Characters with References on Request" or "Understudies," special decimal printout routines, to be stored in 5800-5963. Headings for each of these subroutines are printed by programs in 3932-3963, stored following an expanded operations table. The additional subroutines "outpt," "addr," and "round" fit in 6240-6263 and provide compatible hexadecimal or alphanumeric output, binarization of addresses at a q of 29, and rounding off the fractional portion of numbers at a q of 29, respectively.

PURPOSE

One reason for expanding the already well-integrated Act I system was to help overcome semantics barriers in order that both programmers and scientists unfamiliar with computer terminology would be able to communicate with each other and with the computer without translation of terms outside the computer. The desire for this close relationship developed when an analog computer data display and tape punch system was designed by the author for use in field experiments conducted by the Ocean Sound Propagation Section. Of course, data analysis is only one application of Act I and Encores.

The function of Cast of Characters is to print out the object program in decimal, with optional inclusion of symbols, constants, and operations. The purpose of Understudies is to punch out the object program in compatible decimal instead of hexadecimal in order that it may be moved, for instance, to allow the first eight tracks to be protected in installations having record-inhibit switches. The operations table was expanded to provide non-input typewriter output controls, breakpoint instructions, optional trace of selected locations, a variety of meanings for "-" program input code, and the three additional subroutine symbols "outpt," "addr," and "round."

The Ocean Sound Propagation Section has used a system in the past which includes a convenient alphanumeric output subroutine. Following the policy of improving communications among users, the author added a form of Herring's program (3).

The compatible hexadecimal "outpt" routine is intended to provide more rapid access to punched tape peripheral storage, i.e., to add a link via input and output devices without changing existing subroutines in order to render them compatible. The use of Act III compatible output subroutines was considered; however, necessary rearrangement of the subroutine packages seemed an unwarranted complication.

The "addr" subroutine causes decimal addresses input via the source program to be binarized under control of the object program and program input routine for manipulation in source program terminology.

The "round" operation masks off the 30th and 31st bits of words. It might find use in accurately unpacking fixed point numbers by dividing by a power of 10, rounding, then multiplying them by the same power of 10, at a q of 29.

No "z" delays are employed in these subroutines, permitting a nonstopping, high-speed-punch output mode of operation. Action, however, is slow enough for the typewriter output mode to be equally satisfactory.

ALTERATIONS TO THE COMPILER

In order that the symbol table be generated at 6000 and that nothing be recorded or erased in 5800-5963, the following alterations to the compiler are necessary:

Lo+0012	xh6000	Initialize symbol table
0103	u0104	Bypass adding xz0200 to symbol Lo
0303	u0304	
1038	xz6000	Symbol table Lo
1452	b1038	Inclusion in punchout

When using the altered compiler, dimensions are counted from location 6000 instead of 5800. For this reason the first "dim" statement should contain a number 128 greater than with the standard Act I.

In order that depression of the transfer control button will shift control to headings and, subsequently, to the special decimal memory printout routines rather than again to the compiler immediately following compiling, the 80xt4000 instruction in location 5444 is changed to 80xt3932.

THE OPERATIONS TABLE

The symbol table for Act I and Encores appears in Fig. 1. The resulting operations table compiled by Prelude to Act I and punched by Act I appears in Fig. 2.

No distinction was made between floating point and fixed point calling sequences for logarithm subroutines in order to provide more space in tracks 36 and 37. The left operand should be a number; but if floating log-ln subroutine is employed, the number is not used.

A statement number in the source program should follow "-pir." When unprotected program input routine 10.4 or 10.4L is in memory, the object program stores the location of that statement in the address of the "t" instruction in 0007. Going to program input routine and inputting "-" and seven characters will transfer control to the statement with the seven characters in the accumulator.

"Trace" may be used with or without the transfer control button depressed during compiling. Its intended use is to select printout of intermediate results while computing with the transfer control button depressed.

"CAST OF CHARACTERS WITH REFERENCES ON REQUEST"

The Cast of Characters subroutine is located in 5800-5963 and is under control following the output of its heading program in 3932-3963. Block diagrams of the subroutine appear in Figs. 3 and 4.

After the compiler prints out program and statement locations there is a stop in 5443. If no decimal memory printout is desired at this point, the operator may transfer to another location, or he may turn on the punch, release the transfer control if on, and start to punch the object program at 0300, the symbol table at 6000, and the constants table at 6200.

If a decimal object program printout is desired, the transfer control button should be depressed and the computer started. The output still is to the typewriter, and all breakpoint switches are released. Subsequently, if the transfer control is left down, only the program will be printed, omitting the heading and references. If it is released, the heading and references will be included with the object program printout.

An example of the printout is shown in Fig. 5, followed by the coding of Cast of Characters in Fig. 6. The program in 3932-3963 employs the Cast of Characters alphanumeric subroutine to type out supplementary information, repairs the alphanumeric routine exit, then transfers to 5800 to initialize decimal printout at location 0300.

When the Encores compiler is in memory, a “-” program input routine instruction will transfer to 5802 with the hex address in the accumulator. This address will be the first location printed in decimal. If the address is the same as or greater than the last program location, only one printout cycle will follow. During normal Cast of Characters cycling, the last location and instruction “z0000” will be omitted, and the computer will stop, allowing the operator to choose what he would rather do next. He may either start the punch, use the program in memory, or alter the program first.

The Cast of Characters portion of the subroutine is optimized. The delay instruction following the carriage return is not necessary unless the latter instruction is a tabulation into a long, automatic carriage return, or a stop is desired before punching. When the Reference on Request portion of the subroutine is not to be used, “xz3000” is not needed as a constant and may be changed to “u1805;” otherwise, 5832 may be changed to “xs6331,” Lf, instead of 3000. All temporary storage is within the subroutine. Therefore printout of track 63 contents is possible; however, an attempt to print out contents of tracks 58 or 59 damages the subroutine.

The transfer control may be depressed at any time to abort reference printouts if desired. When it is released, an instruction having an address of 6000-6163 is interpreted as one referring to a symbol. Following printout of the location and instruction, the symbol is typed. If the word found in 6000-6163 happens to contain a non-input (e.g., “0800”) or jam code (e.g., “4800”), that character is bypassed, preventing uncontrolled action, damage, or stoppage. When the computer encounters an instruction having an address of 6200-6239, the word in that location is typed as a five-digit decimal constant at 29. Negative numbers or numbers greater than 99999 will not stop the computer but may contain nondecimal characters. Instruction addresses 3000-5963 are first treated as subroutine bridge locations. If the command is not “u,” as it normally would be following an “r0310” instruction, a “u” – meaning unidentified – is typed. If the command is “u,” the table of transfer addresses at q of 30 in track 37 is searched. If the address is not found, a “u” is printed. When it is found, the symbol in track 36 having the same sector is typed as the operation symbol. For examples, the following instructions would cause the resulting typeouts:

p3200 u	(stop code, not “u” instruction)
z3200 u	(breakpoint, not “u”)
r0305	(address not 3000 or more)
u5000 u	(not transfer address)
b6001 temp.	(symbol)
r0310	(address not 3000 or more)
u4900 asin	(operation symbol)
s6208 04963	(constant)

For future reference, the typewriter punch may be turned on before transferring to Cast of Characters, thereby including the decimal interpretation on the object program tape. Since the program is incapable of punching a conditional stop and the first input word is an eight-character "v" code, the printout will be ignored by the input routines.

"UNDERSTUDIES"

An optional decimal memory printout program may be loaded in 5800-5963 with its headings program in 3932-3963. Understudies (Fig. 7) is similar to Cast of Characters in that it prints locations and instructions in decimal and includes printout of symbols referred to by the object program; however, here the similarity ends.

The Understudies subroutine is headed by loading limits and instructions. The object program, properly modified, can be moved up to the initial region location designated by "dim" statements. The maximum start fill instruction therefore is

$$\text{region Li} - [\text{program Lf} - (\text{program Li} - 1)].$$

The Understudies subroutine does not modify any addresses, so the modifier must be three tracks smaller than the start fill instruction. If subroutines are used with the object program, all return addresses must be changed to program Lo + 0010, the N temporary storage addresses must be changed to program Lo + 0012, and trace subroutine must return to Lo + 0005. Subroutines may be altered by loading a tape of start fill plus instruction sets after the subroutines to be used are loaded and /Lo of program is stored by program input routine.

The Understudies program does not print out constants, operations, or "u." Following carriage return and printout of location, each object program instruction is tested. If the command bits are all zero ("Z"), then ",0000001′ " and the word in hex is printed with a conditional stop. The check sum printout routine of the compiler is employed to output 1 at 2, 1 at 30, breakpoint delay, and stop instructions, since all happen to be positive and contain "0" bits in their command portions. "p" instructions and instructions having addresses greater than object program Lf are typed with (a) leading spaces to clear the remainder of the location out of the accumulator on reloading, (b) an x to prevent modifying, and (c) conditional stop. An instruction having an address 6000-6163 is followed by the symbol in that address. All other instructions are assumed to be modified and are preceded by spaces and no "x." After typing the final instruction, "xz0000," control is immediately transferred to the compiler, which punches the symbol table and constants table on the typewriter in hexadecimal, each with a check sum.

The object program is unchanged in memory, so it may be used immediately or, if desired, punched out in hex at 0300 by transferring to location 5445. Understudies uses the address printout routine in the compiler to type locations and one space, but uses its own address printout routine to type addresses and conditional stop. Hence, interrupting the program cannot alter Act I.

If the object program is to be moved in order that it can be used with protected program input routine, it should be remembered that "outpt," "addr," and "-code" operations then will not work. The first requires subroutines which use constants in unprotected program input routine; the second must "r-u" to program input routine binarization subroutine; and the third must alter an inhibited instruction in order to function.

"OUTPT"-"ADDR"-"ROUND" SUBROUTINE PACKAGES

Since the compiler directly stores symbols of up to five characters at a q of 30 by n-multiplying the six-bit input word, it is convenient to store alphanumeric symbols using the source program. Furthermore, non-input typewriter codes and delays may be assembled by the compiler if appropriate symbols appear in the operations table. The author reviewed a POOL* program by Mr. Herring for alphanumeric output and condensed it for this subroutine package.

Symbol “outpt” for alphanumeric words differs from the alphanumeric output in the Cast of Characters subroutine by omitting protection against non-input codes (Fig. 8). Starting at the left, six bits at a time are shifted into the track portion of a p( ) instruction. p0000 instructions are bypassed, being interpreted as missing character codes of stored symbols having less than five characters. An example of source program and result is:

```
outpt'temp.' '
2'print'temp' '             temp. 662.15 F
ucase'outpt' F' lcase' cr' '
```

If an alphanumeric symbol is not to be replaced by a number in memory before reloading the object program, it should not be identical to the symbol used to represent the number storage location in 6000-6163. A period or space, for instance, might differentiate one from the other.

The “addr” and “round” subroutines help delay the exit from either “outpt” subroutine so that either high-speed punch or typewriter output can be used. The “addr” subroutine is simply a transfer to the binarizing section of 10.4 or 10.4L program input routine and return. Constants and fixed point numbers in the object program are at a q of 29, whereas program input routine codes are at a q of 31. Therefore transfer is made from the “addr” subroutine location to a point beyond the order to n-multiply by 1 at 29. The “round” subroutine contains one extract instruction to mask off bits 30 and 31. Since its transfer address is listed in the operations table, Cast of Characters can include printout of the operation as it can with “addr” and “outpt” operations.

The “addr” and “round” subroutines follow the “outpt” subroutine at the same transfer addresses when combined with the compatible hex “outpt” subroutine (Fig. 9). Four bits at a time are shifted into a p( ) instruction, the hex character identification bit is added, and eight characters – forcing previous contents out of the accumulator – and a conditional stop are printed.

PACKAGING

Act I and Encores have been packaged along with debugging programs, bootstrap with check sum, and supplementary programs to form the following twelve-tape library in the possession of the writer:

I	Prelude, Operations Table, Compiler, Cast of Characters
II	Bootstrap II with Σ, Program Input Routine 10.4L, Index Instructions and Constants, “-” code to Cast of Characters
III	Symbol Table
IV	Alphanumeric/Hexadecimal Subroutines, Bootstrap I
V	Loader and Linkages, Trace-Float-Unfloat, Floating Decimal, Fixed Point Input-Output Subroutines
VI	Floating Point Subroutines
VII	Fixed Point Subroutines
VIII	Program Input Routine 24.2
IX	Decimal Memory Printout and Hex Punch
X	U. of Mich. Trace
XI	React
XII	Understudies

*An LGP-30 user's organization.

React is a one-track assembly (Fig. 10) for use without the compiler to help write efficient programs employing Act I subroutines.

SUMMARY

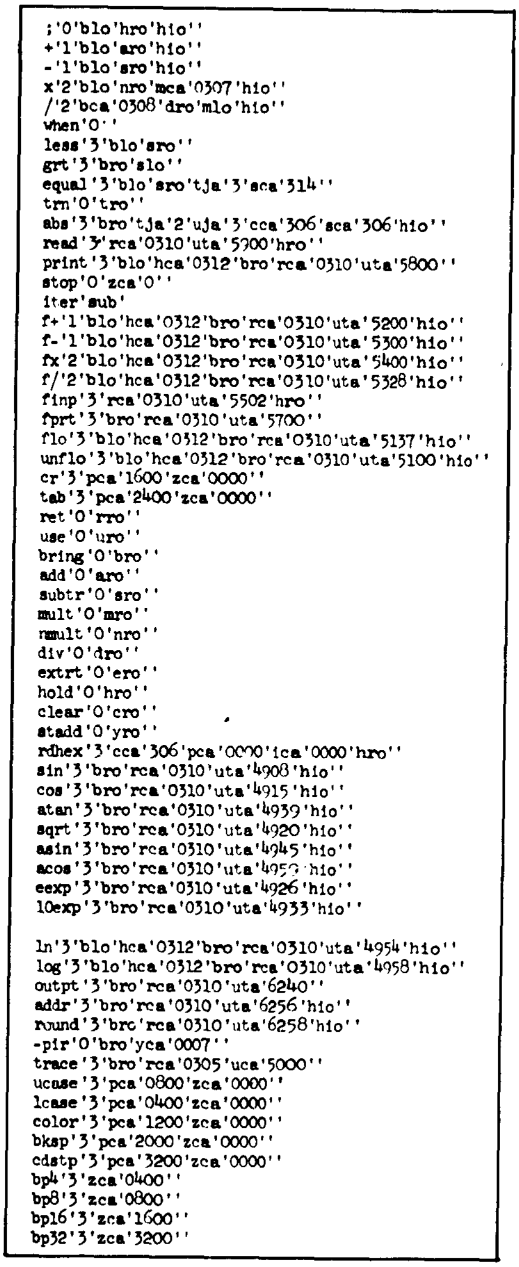
Changes have been made to improve readability and increase the versatility of Act I. No difficulties have been encountered other than remembering to increase the first "dim" statement number by 128 and to refrain from using subroutines requiring 10.4 or 10.4L PIR when protected PIR is used.

REFERENCES

1. Morrisseau, K. C., "Programming the LGP-30 Computer Using Act I," (Code 447) U. S. Navy Bureau of Ships, Washington, D. C.

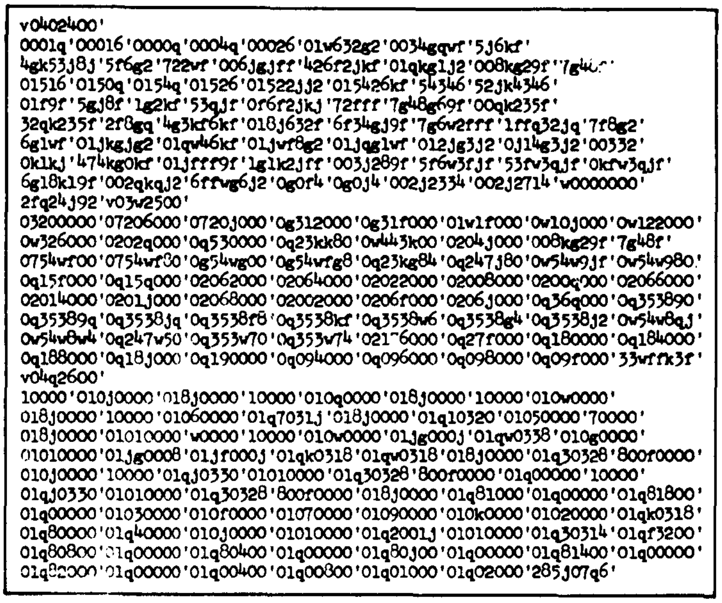
2. Royal McBee Corp., "LGP-30 Act I Compiler," Bulletin No. S-520-R1, Port Chester, N. Y.

3. Herring, R. L., "Alphanumeric Output for Act I," POOL, The LGP-30 Users Organization, Feb. 7, 1961, Pasadena, Calif.



```
;'0'blo'hro'hio''
+'1'blo'aro'hio''
-'1'blo'sro'hio''
x'2'blo'nro'mca'0307'hio''
/'2'bca'0308'dro'mlo'hio''
when'0''
less'3'blo'sro''
grt'3'bro'slo''
equal'3'blo'sro'tja'3'sca'314''
trn'0'tro''
abs'3'bro'tja'2'uja'3'cca'306'sca'306'hio''
read'3'rca'0310'uta'5900'hro''
print'3'blo'hca'0312'bro'rca'0310'uta'5800''
stop'0'zca'0''
iter'sub'
f+'1'blo'hca'0312'bro'rca'0310'uta'5200'hio''
f-'1'blo'hca'0312'bro'rca'0310'uta'5300'hio''
fx'2'blo'hca'0312'bro'rca'0310'uta'5400'hio''
f/'2'blo'hca'0312'bro'rca'0310'uta'5328'hio''
finp'3'rca'0310'uta'5502'hro''
fprt'3'bro'rca'0310'uta'5700''
flo'3'blo'hca'0312'bro'rca'0310'uta'5137'hio''
unflo'3'blo'hca'0312'bro'rca'0310'uta'5100'hio''
cr'3'pca'1600'zca'0000''
tab'3'pca'2400'zca'0000''
ret'0'rro''
use'0'uro''
bring'0'bro''
add'0'aro''
subtr'0'sro''
mult'0'mro''
nmult'0'nro''
div'0'dro''
extrt'0'ero''
hold'0'hro''
clear'0'cro''
stadd'0'yro''
rdhex'3'cca'306'pca'0000'ica'0000'hro''
sin'3'bro'rca'0310'uta'4908'hio''
cos'3'bro'rca'0310'uta'4915'hio''
atan'3'bro'rca'0310'uta'4939'hio''
sqrt'3'bro'rca'0310'uta'4920'hio''
asin'3'bro'rca'0310'uta'4945'hio''
acos'3'bro'rca'0310'uta'4952'hio''
eexp'3'bro'rca'0310'uta'4926'hio''
10exp'3'bro'rca'0310'uta'4933'hio''

ln'3'blo'hca'0312'bro'rca'0310'uta'4954'hio''
log'3'blo'hca'0312'bro'rca'0310'uta'4958'hio''
outpt'3'bro'rca'0310'uta'6240''
addr'3'bro'rca'0310'uta'6256'hio''
round'3'bro'rca'0310'uta'6258'hio''
-pir'0'bro'yca'0007''
trace'3'bro'rca'0305'uca'5000''
ucase'3'pca'0800'zca'0000''
lcase'3'pca'0400'zca'0000''
color'3'pca'1200'zca'0000''
bksp'3'pca'2000'zca'0000''
cdstp'3'pca'3200'zca'0000''
bp4'3'zca'0400''
bp8'3'zca'0800''
bp16'3'zca'1600''
bp32'3'zca'3200''
```

Fig. 1 - "Encores" symbol table



```
v0402400'
0001q'00016'0000q'0004q'00026'01w632g2'0034gqvf'5j6kf'
4gk53j8j'5f6g2'722vf'006jgjff'426f2jkf'01qkg1j2'008kg29f''7g48f'
01516'0150q'0154q'01526'01522jj2'01542 6kf'54346'52jk4346'
01f9f'5gj8f'1g2kf'53qjf'0f6f2jkj'72fff'7g48g69f'00qk235f'
32qk235f'2f8gq'4g3kf6kf'018j632f'6f34gj9f'7g6w2fff'1ffq32jq'7f8g2'
6g1wf'01jkgjg2'01qw46kf'01jwf8g2'01jqg1wf'012jg3j2'0j14g3j2'00332'
0k1kj'474kg0kf'01jfff9f'1g1k2jff'003j289f'5f6w3fjf'53fw3qjf'0kfw3qjf'
6g18k19f'002qkqj2'6ffwg6j2'0g0f4'0g0j4'002j2334'002j2714'w0000000'
2fq24j92'v03w2500'
03200000'07206000'0720j000'0g312000'0g31f000'01w1f000'0w10j000'0w122000'
0w326000'0202q000'0q530000'0q23kk80'0w443k00'0204j000'008kg29f'7g48f'
0754wf00'0754wf80'0g54wg00'0g54wfg8'0q23kg84'0q247j80'0w54w9jf'0w54w980'
0q15f000'0q15q000'02062000'02064000'02022000'02008000'0200q000'02066000'
02014000'0201j000'02068000'02002000'0206f000'0206j000'0q36q000'0q353890'
0q35389q'0q3538jq'0q3538f8'0q3538kf'0q3538w6'0q3538g4'0q3538j2'0w54w8qj'
0w54w8w4'0q247w50'0q353w70'0q353w74'02176000'0q27f000'0q180000'0q184000'
0q188000'0q18j000'0q190000'0q094000'0q096000'0q098000'0q09f000'33wffk3f'
v04q2600'
10000'010j0000'018j0000'10000'010q0000'018j0000'10000'010w0000'
018j0000'10000'01060000'01q7031j'018j0000'01q10320'01050000'70000'
018j0000'01010000'w0000'10000'010w0000'01jg000j'01qw0338'010g0000'
01010000'01jg0008'01jf000j'01qk0318'01qw0318'018j0000'01q30328'800f0000'
010j0000'10000'01qj0330'01010000'01q30328'800f0000'01q00000'10000'
01qj0330'01010000'01q30328'800f0000'018j0000'01q81000'01q00000'01q81800'
01q00000'01030000'010f0000'01070000'01090000'010k0000'01020000'01qk0318'
01q80000'01q40000'010j0000'01010000'01q2001j'01010000'01q30314'01qf3200'
01q80800'01q00000'01q80400'01q00000'01q80j00'01q00000'01q81400'01q00000'
01q82000'01q00000'01q00400'01q00800'01q01000'01q02000'285j07q6'
```

Fig. 2 - "Encores" operation table
]

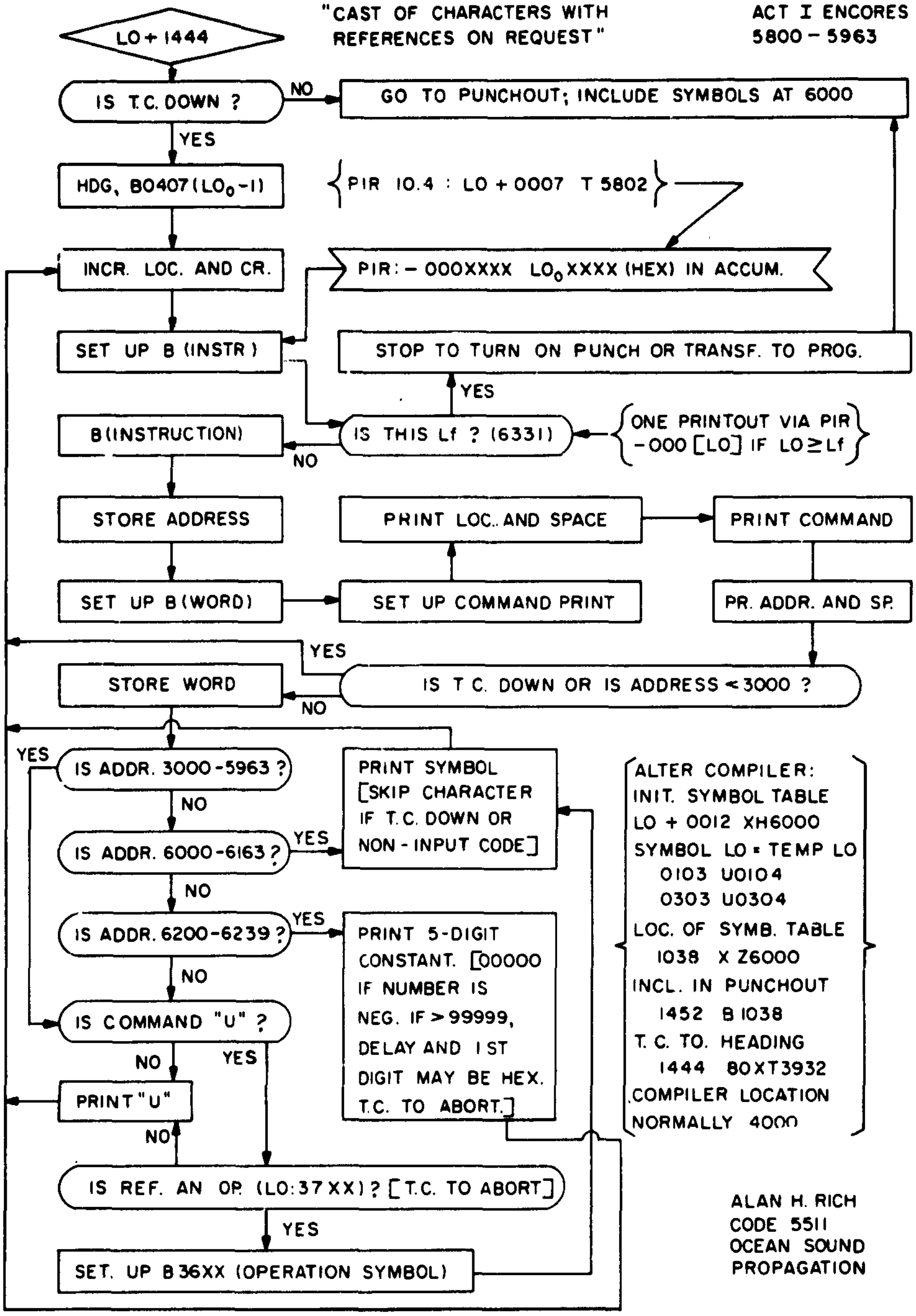
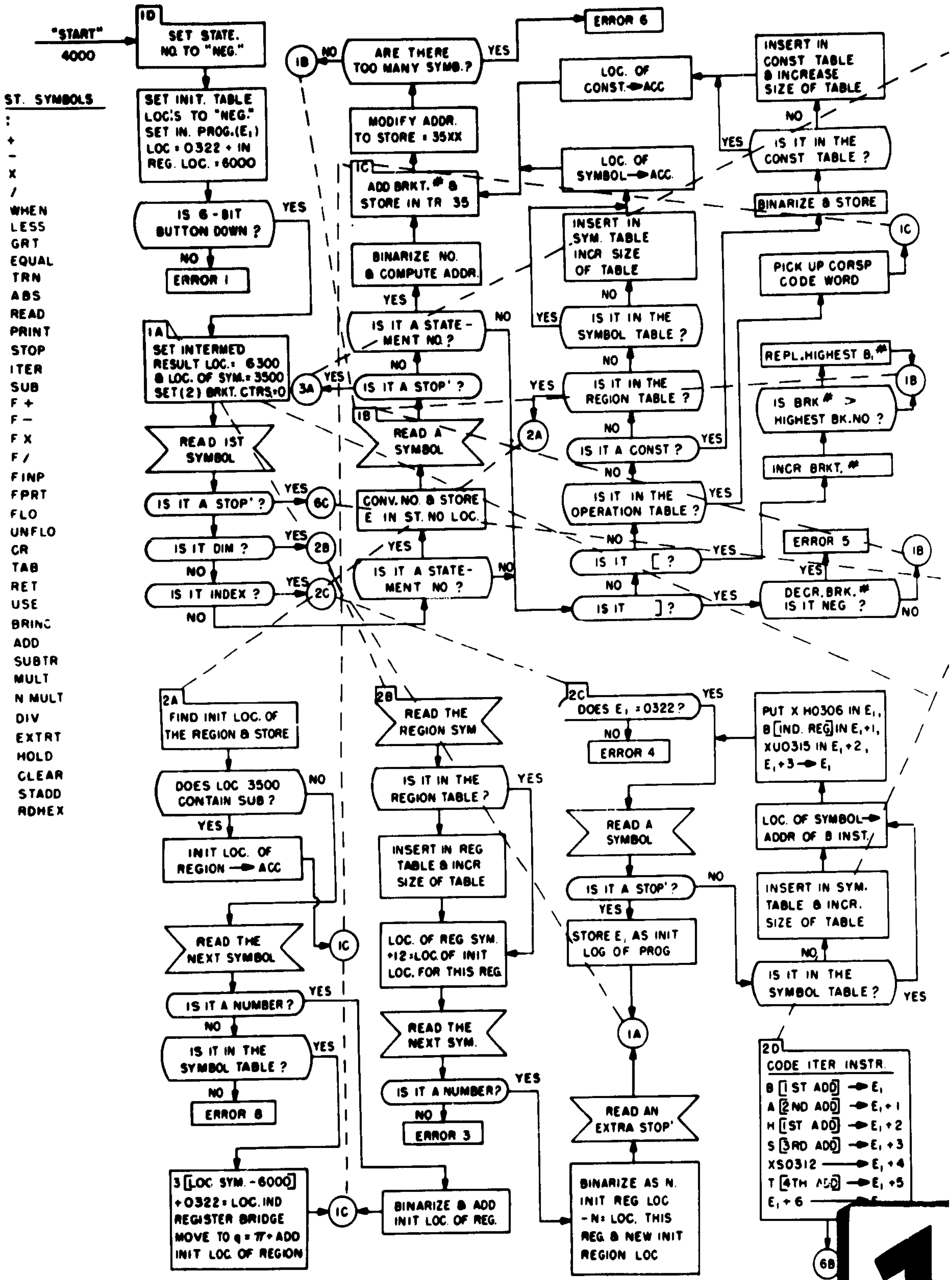


Fig. 3 - "Cast of Characters" block diagram


"START"
4000
ST. SYMBOLS
:
+
-
X
/
WHEN
LESS
GRT
EQUAL
TRN
ABS
READ
PRINT
STOP
ITER
SUB
F +
F −
F X
F /
FINP
FPRT
FLO
UNFLO
CR
TAB
RET
USE
BRINC
ADD
SUBTR
MULT
N MULT
DIV
EXTRT
HOLD
CLEAR
STADD
RDHEX
1D
SET STATE. NO. TO "NEG."
SET INIT. TABLE LOC'S TO "NEG." SET IN. PROG.(E1) LOC = 0322 + IN REG. LOC. = 6000
IS 6-BIT BUTTON DOWN ?
YES
NO
ERROR 1
1A
SET INTERMED RESULT LOC. = 6300 & LOC. OF SYM. = 3500 SET (2) BRKT. CTRS = 0
READ 1ST SYMBOL
IS IT A STOP' ?
YES
6C
IS IT DIM ?
YES
2B
NO
IS IT INDEX ?
YES
2C
NO
1B
NO
ARE THERE TOO MANY SYMB. ?
YES
ERROR 6
MODIFY ADDR. TO STORE = 35XX
1C
ADD BRKT. # & STORE IN TR 35
BINARIZE NO. & COMPUTE ADDR.
YES
IS IT A STATEMENT NO. ?
NO
NO
3A
YES
IS IT A STOP' ?
1B
READ A SYMBOL
2A
CONV. NO. & STORE E IN ST. NO LOC.
YES
IS IT A STATEMENT NO ?
NO
LOC. OF CONST. → ACC
LOC. OF SYMBOL → ACC.
INSERT IN SYM. TABLE INCR SIZE OF TABLE
NO
YES
IS IT IN THE SYMBOL TABLE ?
NO
YES
IS IT IN THE REGION TABLE ?
NO
IS IT A CONST ?
YES
NO
IS IT IN THE OPERATION TABLE ?
YES
NO
IS IT [?
YES
NO
IS IT] ?
YES
INSERT IN CONST TABLE & INCREASE SIZE OF TABLE
NO
YES
IS IT IN THE CONST TABLE ?
BINARIZE & STORE
1C
PICK UP CORSP CODE WORD
REPL. HIGHEST B. #
1B
IS BRK # > HIGHEST BK. NO ?
INCR BRKT. #
ERROR 5
1B
YES
DECR. BRK. # IS IT NEG ?
NO
2A
FIND INIT LOC. OF THE REGION & STORE
DOES LOC 3500 CONTAIN SUB ?
NO
YES
INIT LOC. OF REGION → ACC
1C
READ THE NEXT SYMBOL
IS IT A NUMBER ?
YES
NO
IS IT IN THE SYMBOL TABLE ?
YES
NO
ERROR 8
3 [LOC SYM. − 6000] + 0322 = LOC. IND REGISTER BRIDGE MOVE TO q = π + ADD INIT LOC OF REGION
1C
2B
READ THE REGION SYM
IS IT IN THE REGION TABLE ?
YES
INSERT IN REG TABLE & INCR SIZE OF TABLE
LOC. OF REG SYM. + 12 = LOC. OF INIT LOC. FOR THIS REG.
READ THE NEXT SYM.
IS IT A NUMBER ?
YES
NO
ERROR 3
BINARIZE & ADD INIT LOC. OF REG.
2C
DOES E1 = 0322 ?
YES
NO
ERROR 4
READ A SYMBOL
IS IT A STOP' ?
NO
YES
STORE E1 AS INIT LOG OF PROG
1A
READ AN EXTRA STOP'
BINARIZE AS N. INIT REG LOC − N = LOC. THIS REG & NEW INIT REGION LOC
PUT X H0306 IN E1, B [IND. REG] IN E1+1, XU0315 IN E1+2, E1+3 → E1
LOC. OF SYMBOL → ADDR OF B INST.
INSERT IN SYM. TABLE & INCR. SIZE OF TABLE
NO
IS IT IN THE SYMBOL TABLE ?
YES
2D
CODE ITER INSTR.
B [1ST ADD] → E1
A [2ND ADD] → E1+1
H [1ST ADD] → E1+2
S [3RD ADD] → E1+3
XS0312 → E1+4
T [4TH ADD] → E1+5
E1+6 →
6B

1

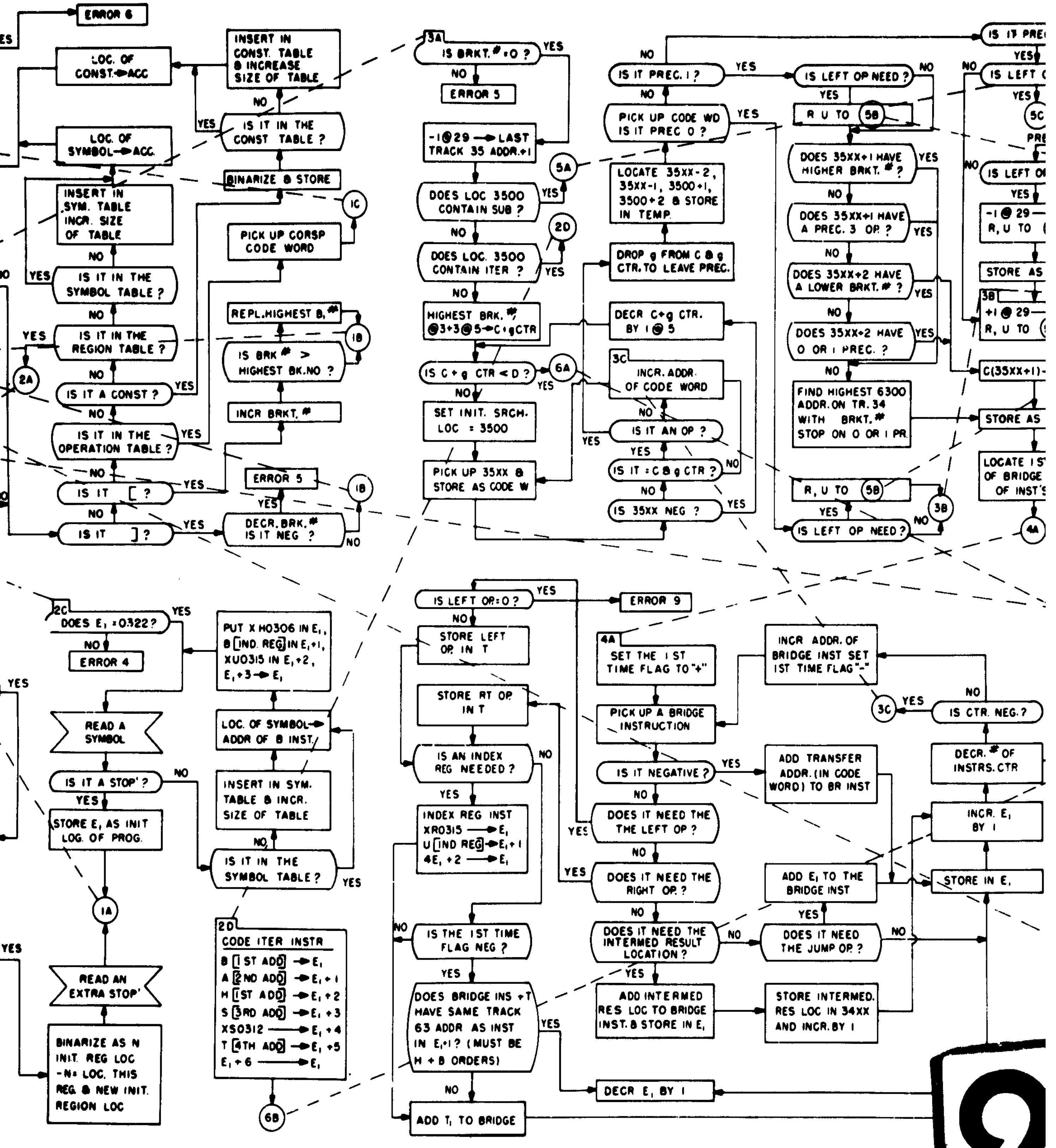
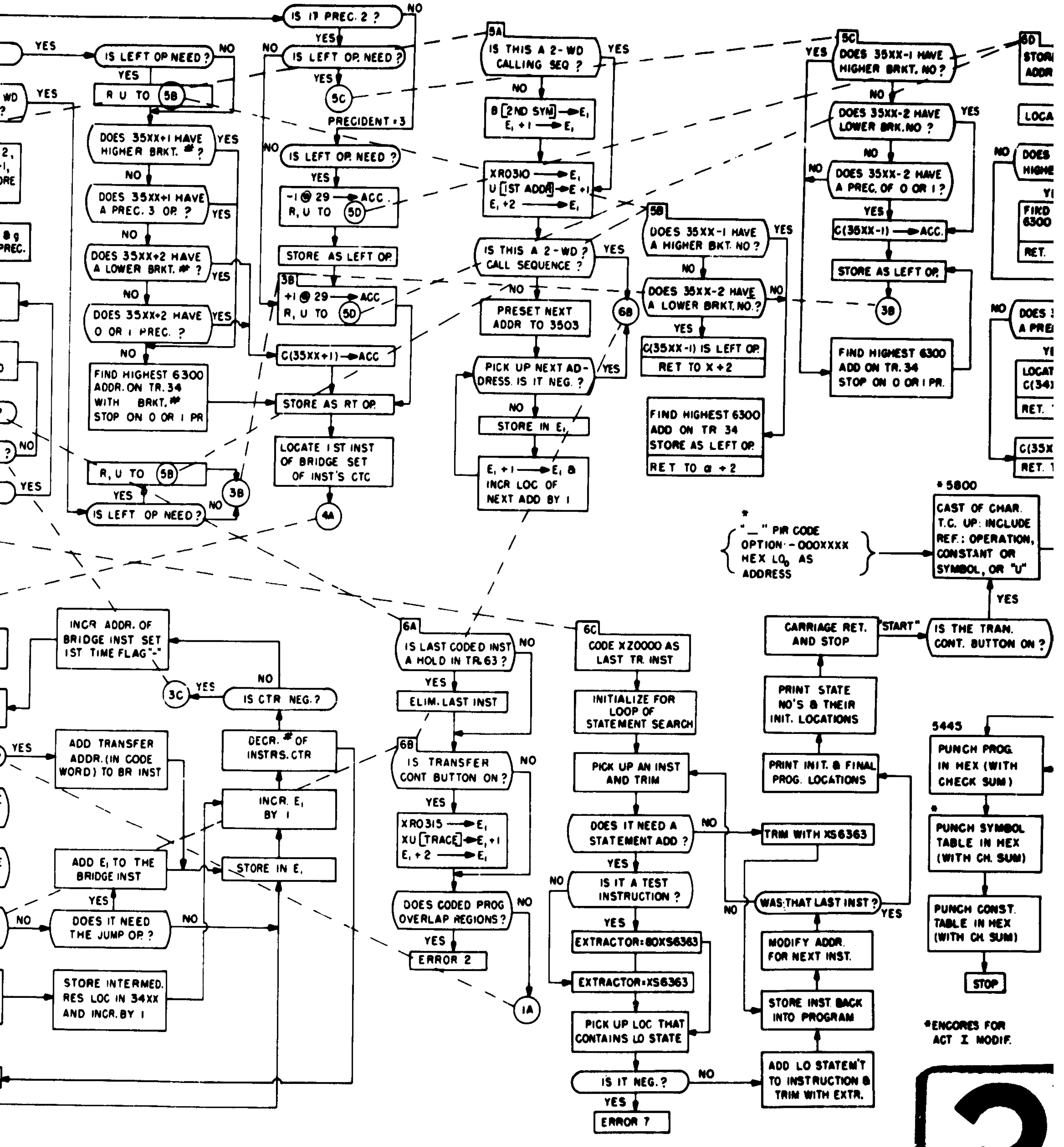
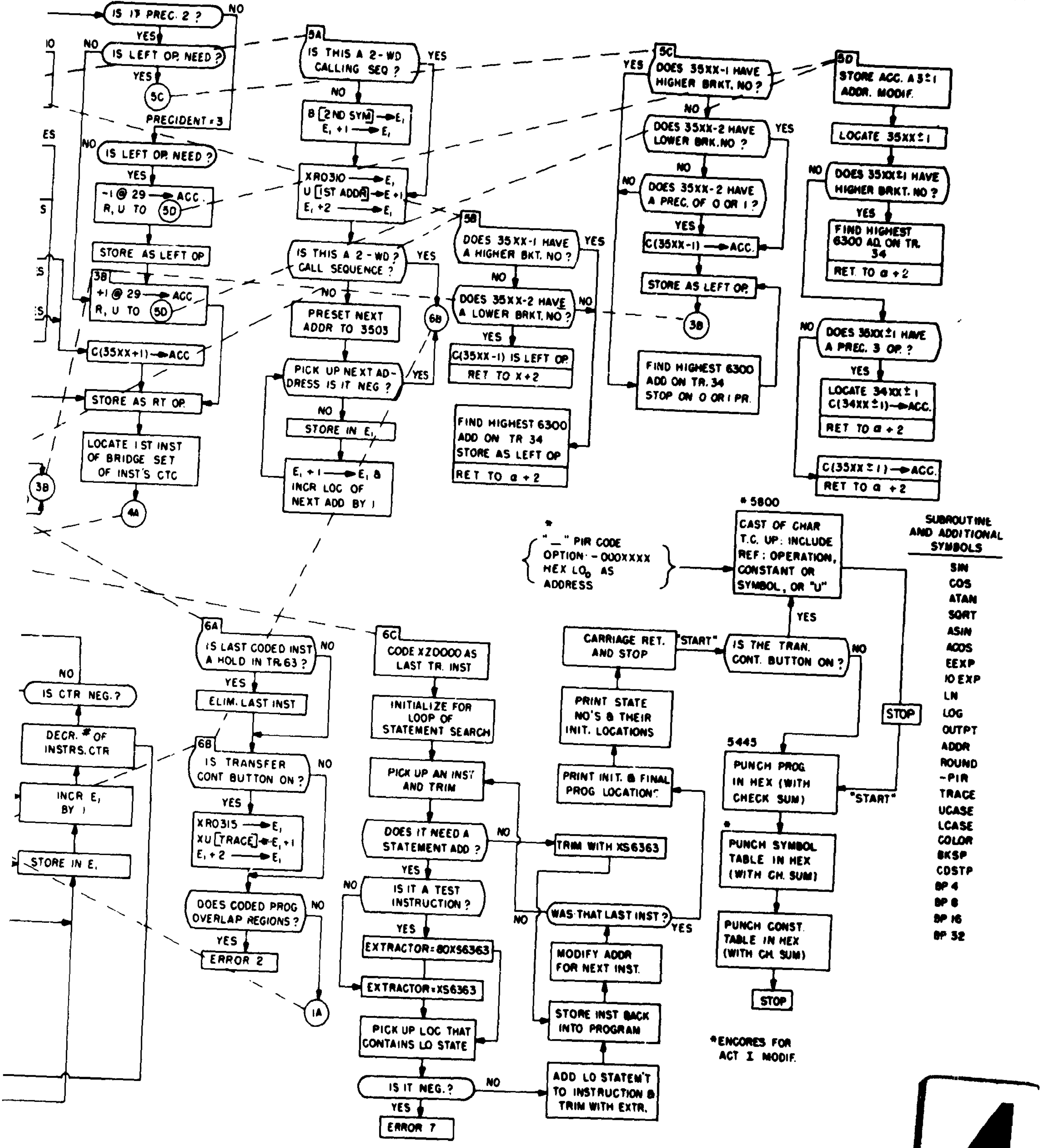

ERROR 6
LOC. OF CONST.→ACC
INSERT IN CONST. TABLE & INCREASE SIZE OF TABLE
IS IT IN THE CONST TABLE ?
LOC. OF SYMBOL→ACC.
BINARIZE & STORE
INSERT IN SYM. TABLE INCR. SIZE OF TABLE
PICK UP CORSP CODE WORD
IS IT IN THE SYMBOL TABLE ?
REPL.HIGHEST B.#
IS IT IN THE REGION TABLE ?
IS BRK # > HIGHEST BK.NO ?
IS IT A CONST ?
INCR BRKT.#
IS IT IN THE OPERATION TABLE ?
ERROR 5
IS IT [?
IS IT] ?
DECR.BRK.# IS IT NEG ?
1C
1B
2A
3A IS BRKT.# =0 ?
ERROR 5
-1@29 → LAST TRACK 35 ADDR.+1
DOES LOC 3500 CONTAIN SUB ?
DOES LOC. 3500 CONTAIN ITER ?
HIGHEST BRK.# @3+3@5→C+gCTR
IS C+g CTR < D ?
SET INIT. SRCH. LOC = 3500
PICK UP 35XX & STORE AS CODE W
5A
2D
6A
IS IT PREC. 1 ?
PICK UP CODE WD IS IT PREC 0 ?
LOCATE 35XX-2, 35XX-1, 3500+1, 3500+2 & STORE IN TEMP.
DROP g FROM C & g CTR.TO LEAVE PREC.
DECR C+g CTR. BY 1 @ 5
3C INCR. ADDR. OF CODE WORD
IS IT AN OP ?
IS IT =C & g CTR ?
IS 35XX NEG ?
IS LEFT OP NEED ?
R U TO 5B
DOES 35XX+1 HAVE HIGHER BRKT.# ?
DOES 35XX+1 HAVE A PREC. 3 OP. ?
DOES 35XX+2 HAVE A LOWER BRKT.# ?
DOES 35XX+2 HAVE 0 OR 1 PREC. ?
FIND HIGHEST 6300 ADDR.ON TR. 34 WITH BRKT.# STOP ON 0 OR 1 PR.
R, U TO 5B
3B
IS LEFT OP NEED ?
IS IT PREC
IS LEFT O
5C
PRE
IS LEFT OI
-1 @ 29 — R, U TO
STORE AS
3B +1 @ 29— R, U TO
C(35XX+1)-
STORE AS
LOCATE 1 ST OF BRIDGE OF INST'S
4A
2C DOES E₁ =0322?
ERROR 4
PUT X H0306 IN E₁, B [IND. REG] IN E₁+1, XU0315 IN E₁+2, E₁+3 → E₁
READ A SYMBOL
LOC. OF SYMBOL→ ADDR OF B INST.
IS IT A STOP' ?
INSERT IN SYM. TABLE & INCR. SIZE OF TABLE
STORE E₁ AS INIT LOG. OF PROG.
IS IT IN THE SYMBOL TABLE ?
1A
2D CODE ITER INSTR
B [1ST ADD] → E₁
A [2ND ADD] → E₁+1
H [1ST ADD] → E₁+2
S [3RD ADD] → E₁+3
XS0312 → E₁+4
T [4TH ADD] → E₁+5
E₁+6 → E₁
READ AN EXTRA STOP'
BINARIZE AS N INIT. REG LOC -N= LOC. THIS REG. & NEW INIT. REGION LOC
6B
IS LEFT OP.=0 ?
ERROR 9
STORE LEFT OP. IN T
STORE RT OP. IN T
IS AN INDEX REG NEEDED ?
INDEX REG INST XR0315 → E₁ U [IND REG] → E₁+1 4E₁ +2 → E₁
IS THE 1ST TIME FLAG NEG ?
DOES BRIDGE INS +T HAVE SAME TRACK 63 ADDR AS INST IN E₁+1? (MUST BE H + B ORDERS)
ADD T₁ TO BRIDGE
4A SET THE 1ST TIME FLAG TO "+"
PICK UP A BRIDGE INSTRUCTION
IS IT NEGATIVE ?
DOES IT NEED THE THE LEFT OP ?
DOES IT NEED THE RIGHT OP. ?
DOES IT NEED THE INTERMED RESULT LOCATION ?
ADD INTERMED RES LOC TO BRIDGE INST. & STORE IN E₁
DECR E₁ BY 1
INCR ADDR. OF BRIDGE INST SET 1ST TIME FLAG "-"
ADD TRANSFER ADDR. (IN CODE WORD) TO BR INST
ADD E₁ TO THE BRIDGE INST
DOES IT NEED THE JUMP OP. ?
STORE INTERMED. RES LOC IN 34XX AND INCR. BY 1
3C
IS CTR. NEG.?
DECR. # OF INSTRS. CTR
INCR. E₁ BY 1
STORE IN E₁
YES
NO

Fig. 4 - Altered "Act I" block diagram

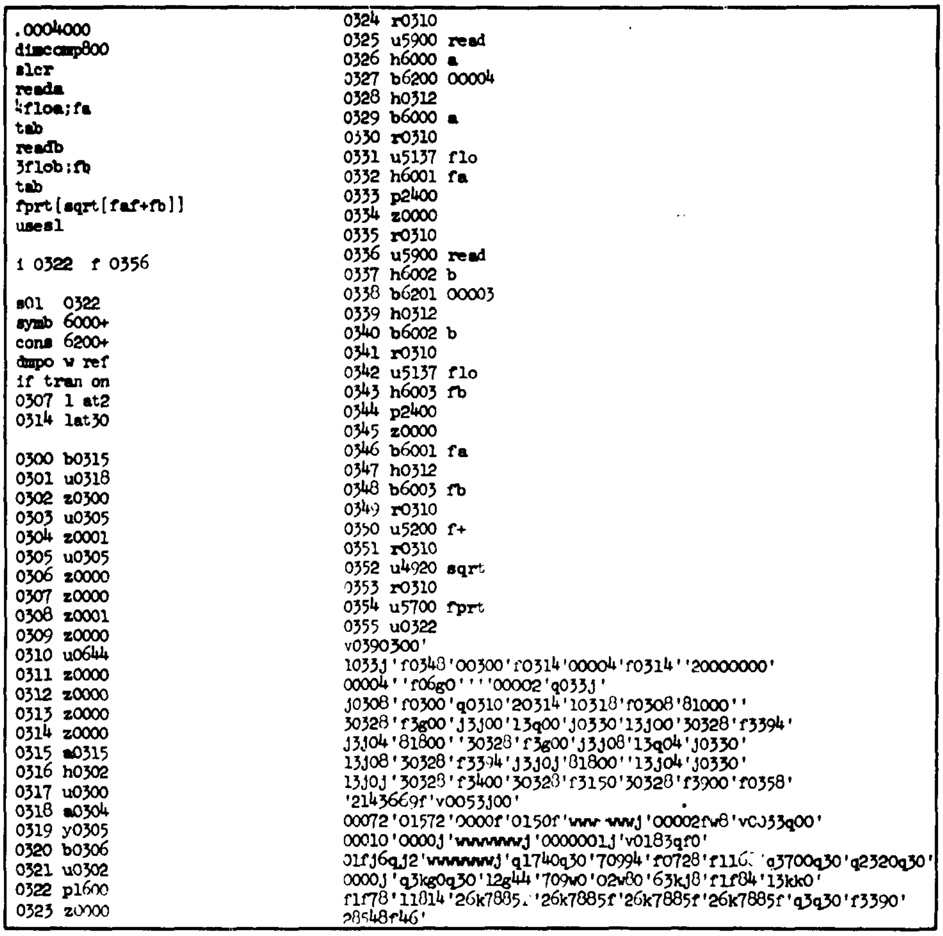

IS IT PREC. 2 ?
IS LEFT OP NEED ?
R U TO 5B
DOES 35XX+1 HAVE HIGHER BRKT. # ?
DOES 35XX+1 HAVE A PREC. 3 OR ?
DOES 35XX+2 HAVE A LOWER BRKT. # ?
DOES 35XX+2 HAVE 0 OR 1 PREC. ?
FIND HIGHEST 6300 ADDR. ON TR. 34 WITH BRKT. # STOP ON 0 OR 1 PR
R, U TO 5B
IS LEFT OP NEED ?
IS LEFT OP. NEED ?
5C
PRECIDENT = 3
IS LEFT OP. NEED ?
-1 @ 29 → ACC. R, U TO 5D
STORE AS LEFT OP.
3B
+1 @ 29 → ACC R, U TO 5D
C(35XX+1) → ACC
STORE AS RT OP.
LOCATE 1ST INST OF BRIDGE SET OF INST'S CTC
3B
4A
5A
IS THIS A 2-WD CALLING SEQ ?
B [2ND SYM] → E₁ E₁ + 1 → E₁
XR0310 → E₁ U [1ST ADDR] → E₁+1 E₁ +2 → E₁
IS THIS A 2-WD ? CALL SEQUENCE ?
PRESET NEXT ADDR TO 3503
PICK UP NEXT ADDRESS. IS IT NEG. ?
STORE IN E₁
E₁ +1 → E₁ & INCR LOC OF NEXT ADD BY 1
6B
5B
DOES 35XX-1 HAVE A HIGHER BKT. NO ?
DOES 35XX-2 HAVE A LOWER BRKT. NO.?
C(35XX-1) IS LEFT OP.
RET TO X+2
FIND HIGHEST 6300 ADD ON TR 34 STORE AS LEFT OP.
RET TO α+2
5C
DOES 35XX-1 HAVE HIGHER BRKT. NO ?
DOES 35XX-2 HAVE LOWER BRK. NO ?
DOES 35XX-2 HAVE A PREC. OF 0 OR 1 ?
C(35XX-1) → ACC.
STORE AS LEFT OP.
3B
FIND HIGHEST 6300 ADD ON TR. 34 STOP ON 0 OR 1 PR.
5D
* 5800
CAST OF CHAR. T.C. UP: INCLUDE REF.: OPERATION, CONSTANT OR SYMBOL, OR "U"
* "—" PIR CODE OPTION - 000XXXX HEX LO₀ AS ADDRESS
IS THE TRAN. CONT. BUTTON ON ?
"START"
CARRIAGE RET. AND STOP
PRINT STATE NO'S & THEIR INIT. LOCATIONS
PRINT INIT. & FINAL PROG. LOCATIONS
5445
PUNCH PROG IN HEX (WITH CHECK SUM)
* PUNCH SYMBOL TABLE IN HEX (WITH CH. SUM)
PUNCH CONST. TABLE IN HEX (WITH CH. SUM)
STOP
*ENCORES FOR ACT I MODIF.
INCR ADDR. OF BRIDGE INST SET 1ST TIME FLAG "-"
3C
IS CTR NEG. ?
DECR. # OF INSTRS. CTR
ADD TRANSFER ADDR. (IN CODE WORD) TO BR INST
INCR. E₁ BY 1
ADD E₁ TO THE BRIDGE INST
STORE IN E₁
DOES IT NEED THE JUMP OP. ?
STORE INTERMED. RES LOC IN 34XX AND INCR. BY 1
6A
IS LAST CODED INST A HOLD IN TR. 63 ?
ELIM. LAST INST
6B
IS TRANSFER CONT BUTTON ON ?
XR0315 → E₁ XU [TRACE] → E₁+1 E₁ + 2 → E₁
DOES CODED PROG OVERLAP REGIONS ?
ERROR 2
1A
6C
CODE XZ0000 AS LAST TR. INST
INITIALIZE FOR LOOP OF STATEMENT SEARCH
PICK UP AN INST AND TRIM
DOES IT NEED A STATEMENT ADD ?
IS IT A TEST INSTRUCTION ?
EXTRACTOR = 80XS6363
EXTRACTOR = XS6363
PICK UP LOC THAT CONTAINS LO STATE
IS IT NEG. ?
ERROR 7
TRIM WITH XS6363
WAS THAT LAST INST ?
MODIFY ADDR. FOR NEXT INST.
STORE INST. BACK INTO PROGRAM
ADD LO STATEM'T TO INSTRUCTION & TRIM WITH EXTR.
YES
NO

Act I" block diagram

3


IS 17 PREC. 2 ?
NO
YES
IS LEFT OP. NEED ?
NO
YES
5C
PRECIDENT = 3
IS LEFT OP. NEED ?
NO
YES
-1 @ 29 → ACC.
R, U TO 5D
STORE AS LEFT OP
3B
+1 @ 29 → ACC
R, U TO 5D
C(35XX+1) → ACC
STORE AS RT OP
LOCATE 1ST INST OF BRIDGE SET OF INST'S CTC
3B
4A
5A
IS THIS A 2-WD CALLING SEQ ?
YES
NO
B [2ND SYM] → E₁
E₁ + 1 → E₁
XR0310 → E₁
U [1ST ADDR] → E + 1
E₁ + 2 → E₁
IS THIS A 2-WD CALL SEQUENCE ?
YES
NO
PRESET NEXT ADDR TO 3503
6B
PICK UP NEXT ADDRESS IS IT NEG ?
YES
NO
STORE IN E₁
E₁ + 1 → E₁ & INCR LOC OF NEXT ADD BY 1
5B
DOES 35XX-1 HAVE A HIGHER BKT. NO ?
YES
NO
DOES 35XX-2 HAVE A LOWER BRKT. NO ?
NO
YES
C(35XX-1) IS LEFT OP
RET TO X+2
FIND HIGHEST 6300 ADD ON TR 34 STORE AS LEFT OP
RET TO α + 2
5C
YES
DOES 35XX-1 HAVE HIGHER BRKT. NO ?
NO
DOES 35XX-2 HAVE LOWER BRK. NO ?
YES
NO
NO
DOES 35XX-2 HAVE A PREC. OF 0 OR 1 ?
YES
C(35XX-1) → ACC.
STORE AS LEFT OP.
3B
FIND HIGHEST 6300 ADD ON TR. 34 STOP ON 0 OR 1 PR.
5D
STORE ACC. A 3±1 ADDR. MODIF.
LOCATE 35XX±1
NO
DOES 35XX±1 HAVE HIGHER BRKT. NO ?
YES
FIND HIGHEST 6300 AD. ON TR. 34
RET. TO α + 2
NO
DOES 35XX±1 HAVE A PREC. 3 OP. ?
YES
LOCATE 34XX ± 1 C(34XX ± 1) → ACC.
RET TO α + 2
C(35XX ± 1) → ACC.
RET TO α + 2
*
"_" PIR CODE OPTION - 000XXXX HEX LO₀ AS ADDRESS
* 5800
CAST OF CHAR T.C. UP: INCLUDE REF: OPERATION, CONSTANT OR SYMBOL, OR "U"
YES
CARRIAGE RET. AND STOP
"START"
IS THE TRAN. CONT. BUTTON ON ?
NO
STOP
PRINT STATE NO'S & THEIR INIT. LOCATIONS
PRINT INIT. & FINAL PROG LOCATIONS
5445
PUNCH PROG IN HEX (WITH CHECK SUM)
"START"
*
PUNCH SYMBOL TABLE IN HEX (WITH CH. SUM)
PUNCH CONST. TABLE IN HEX (WITH CH. SUM)
STOP
SUBROUTINE AND ADDITIONAL SYMBOLS
SIN
COS
ATAN
SQRT
ASIN
ACOS
EEXP
10 EXP
LN
LOG
OUTPT
ADDR
ROUND
-PIR
TRACE
UCASE
LCASE
COLOR
BKSP
CDSTP
BP 4
BP 8
BP 16
BP 32
*ENCORES FOR ACT I MODIF.
NO
IS CTR NEG.?
DECR. # OF INSTRS. CTR
INCR E₁ BY 1
STORE IN E₁
6A
IS LAST CODED INST A HOLD IN TR. 63 ?
NO
YES
ELIM. LAST INST
6B
IS TRANSFER CONT BUTTON ON ?
NO
YES
XR0315 → E₁
XU [TRACE] → E₁ + 1
E₁ + 2 → E₁
DOES CODED PROG OVERLAP REGIONS ?
NO
YES
ERROR 2
1A
6C
CODE XZ0000 AS LAST TR. INST
INITIALIZE FOR LOOP OF STATEMENT SEARCH
PICK UP AN INST AND TRIM
DOES IT NEED A STATEMENT ADD ?
NO
YES
TRIM WITH XS6363
NO
IS IT A TEST INSTRUCTION ?
YES
EXTRACTOR = 80XS6363
EXTRACTOR = XS6363
PICK UP LOC THAT CONTAINS LO STATE
IS IT NEG. ?
NO
YES
ERROR 7
NO
WAS THAT LAST INST ?
YES
MODIFY ADDR FOR NEXT INST.
STORE INST BACK INTO PROGRAM
ADD LO STATEM'T TO INSTRUCTION & TRIM WITH EXTR.
10
ES
S
S
S

4



```
.0004000
dimcomp800
slcr
reada
4floa;fa
tab
readb
3flob;fb
tab
fprt[sqrt[faf+fb]]
usesl

i 0322  f 0356

s01  0322
symb 6000+
cons 6200+
dmpo w ref
if tran on
0307 1 at2
0314 lat30

0300 b0315
0301 u0318
0302 z0300
0303 u0305
0304 z0001
0305 u0305
0306 z0000
0307 z0000
0308 z0001
0309 z0000
0310 u0644
0311 z0000
0312 z0000
0313 z0000
0314 z0000
0315 a0315
0316 h0302
0317 u0300
0318 a0304
0319 y0305
0320 b0306
0321 u0302
0322 p1600
0323 z0000
```

```
0324 r0310
0325 u5900 read
0326 h6000 a
0327 b6200 00004
0328 h0312
0329 b6000 a
0330 r0310
0331 u5137 flo
0332 h6001 fa
0333 p2400
0334 z0000
0335 r0310
0336 u5900 read
0337 h6002 b
0338 b6201 00003
0339 h0312
0340 b6002 b
0341 r0310
0342 u5137 flo
0343 h6003 fb
0344 p2400
0345 z0000
0346 b6001 fa
0347 h0312
0348 b6003 fb
0349 r0310
0350 u5200 f+
0351 r0310
0352 u4920 sqrt
0353 r0310
0354 u5700 fprt
0355 u0322
v0390300'
1033j'f0348'00300'f0314'00004'f0314''20000000'
00004''f06g0''''00002'q033j'
j0308'f0300'q0310'20314'10318'f0308'81000''
30328'f3g00'j3j00'13q00'j0330'13j00'30328'f3394'
j3j04'81800''30328'f3g00'j3j08'13q04'j0330'
13j08'30328'f3394'j3j0j'81800''13j04'j0330'
13j0j'30328'f3400'30328'f3150'30328'f3900'f0358'
'2143669f'v0053j00'
00072'01572'0000f'0150f'www wwj'00002fw8'v0033q00'
00010'0000j'wwwwwwj'0000001j'v0183qf0'
01fj6qj2'wwwwwwj'q1740q30'70994'f0728'f116''q3700q30'q2320q30'
0000j'q3kg0q30'12g44'709w0'02w80'63kj8'f1f84'13kk0'
f1f78'11814'26k7885f'26k7885f'26k7885f'26k7885f'q3q30'f3390'
28548f46'
```

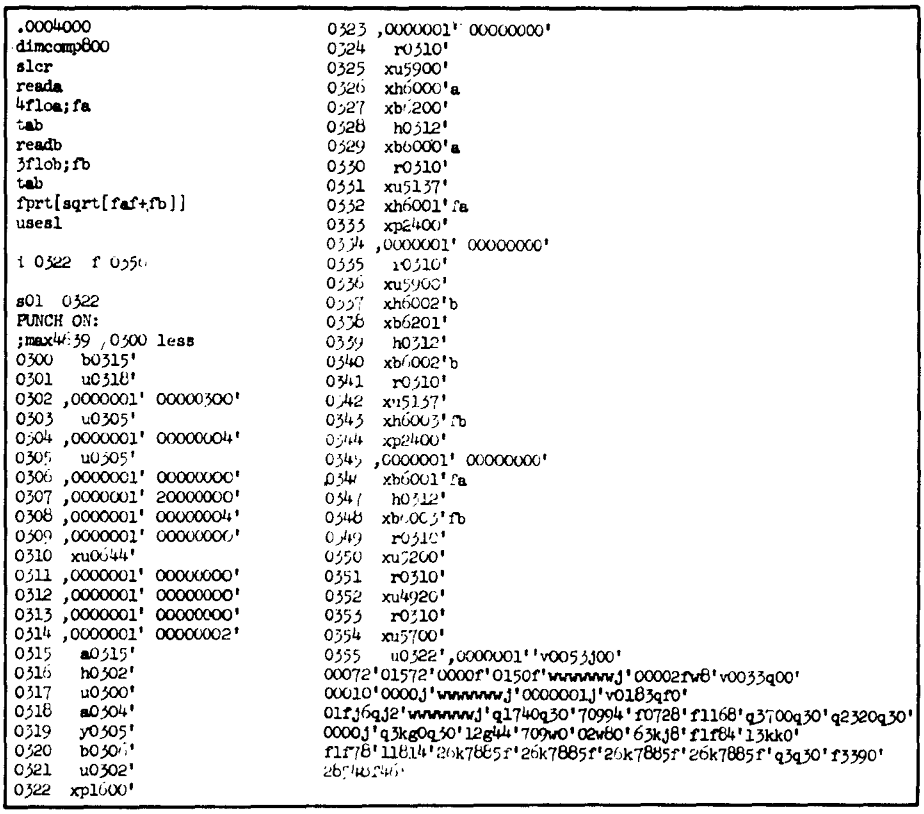
Fig. 5 - "Cast of Characters" printout

```
;0000007'     xu5802'  "-" [illegible]

;0003932'
,0003900'     32     z0051'  stop to rel. t.c.
              33     b0032'
              34     y0037'  init. b[  ]
              35     a0047'  1 at 29
              36     y0041'  b[  ]
              37     bzzzz'  odd word
              38     t0049'  finished?
              39    xr5963'
              40    xu5944'  alpha output
              41     bzzzz'  even word
              42    xr5963'
              43    xu5944'  alpha output
              44    xp1623'  carriage ret.
              45     b0041'  b[  ]
              46     a0047'  1 at 29
              47    xz0001'  delay
              48     u0034'  loop
              49    xy5963'  repair Cast
              50    xu5800'  to Cast
,0000012'     51  7f4gf286'  "symb "
              52  34104116'  "6000+"
              53  6g1g3q86'  "cons "
              54  34504116'  "6200+"
              55  2fqj3186'  "dmpo "
              56  7j19g2k4'  "w ref"
              57  2350769f'  "if tr"
              58  72j871g2'  "an on"
              59  04704w06'  "0307 "
              60  0j1w3694'  "1 at2"
              61  0470j906'  "0314 ".
              62  0kjkf704'  "1at30"
              63  80xt5801'  flag + repair

;0004012'           xh6000'  init. symbols

;0004103'           xu4104'  bypass move

;0004303'           xu4304'  bypass move

;0005038'           xz6000'  Lo symbol table

;0005452'           xb5038'  punch symbols

;0005444'         80xt3932'  Cast headings
```

```
;0005800'
'0004000'   1800     b0407'  xz0263
              01     a1830'  1 at 29
              02     y1817'  b(instruction)
              03    xp1653'  cr. or tab into cr.
v1805 or      04    xz3000'  delay
              05     y1841'  location
              06    xs6331'  obj. program Lf
              07     t1817'  not finished?
,0000001'     08   2000000'  stop
              09     u1445'  to hex punch
              10    xz0100'  command tag
              11     pzzzz'  command character
              12    xz0040'  delay
              13     b1840'  address
              14     r1745'
              15     u1737'  print addr + space
              16     u1831'  to test address
              17     bzzzz'  instruction
              18     y1840'  address
              19     y1834'  b[  ] word
              20     u1822'
              21  [        ]'  word at address
              22     m1808'  1 at 6
              23     e1838'  xz6000
              24     a1810'  xz0100
              25     y1811'  set up command
              26     b1841'  location
              27     r1745'
              28     u1737'  print loc + space
              29     u1811'  to print instr
              30    xz0001'
              31     b1840'  address
xs6331 or     32     s1804'  xz3000
              33  800t1962'  abort under 3000?
              34     bzzzz'  symbol
              35     h1821'  store
              36     u1842'  to test address
              37    xp4100'  "u" (unidentified)
              38    xz6000'  delay
              39     u1902'  to incr location
              40     zzzzz'  address
              41     zzzzz'  location
              42     b1840'  address
              43     s1838'  xz6000
              44     t1849'  3000-5963 SUB?
              45     s1915'  xz0200
              46     t1945'  6000-6163 SYMBOL?
              47     s1812'  xz0040
              48     t1860'  6200-6239 CONSTANT?
              49     b1811'  command
              50     s1837'  xp4100
              51     t1837'  not "u": not sub
              52     s1810'  xz0100
              53     t1916'  yes: SEARCH OPS.
              54     u1837'  no: print "u"
              55    xz0001'  1 at 29
              56    xz0010'  10
              57    xz0136'  100
              58    xz1540'  1000
,0000001'     59      9j40'  10000
              60     b1939'  s1859 CONSTANT
              61     h1903'
              62     s1931'  s1855
              63  800t1962'  abort/5 digits?
```

```
            1900     b1915'  xz0200
              01     y1909'  p[  ]
              02     b1821'  word
              03  [        ]'  s(EXP)/alpha ctr.
              04     t1909'  print digit?
              05     h1821'  remainder
              06     b1909'  p[  ]
              07     a1933'  xz0400 (Δ"1")
              08     u1901'  retest
              09     pzzzz'  print digit
              10     b1903'  s[  ]
              11    xz6363'  delay
              12     s1855'  1 at 29
              13     u1861'  set up s(EXP)
              14    xb3700'  dummy instruction
              15    xz0200'
              16     b1914'  xb4700 SEARCH OPS.
              17     c1918'  trial op code
              18  [        ]'  b(Lo of code + 1 tr.)
              19     n1955'  1 at 30
              20     e1911'  xz6363
              21     s1840'  address
              22     t1925'  not addr: incr b[  ]
              23     s1830'  1 at 29
              24  800t1940'  abort/set up symbol
              25     b1918'  no: incr b[  ]
              26     a1855'  1 at 29
              27     y1918'  b[  ]
              28     s1932'  xb3763 search Lf
              29     t1918'  to test next op
              30     u1837'  not in table: "u"
              31     s1855'  dummy instr
              32    xb3763'  dummy instr
              33    xz0400'  Δ"1"
,0000001'     34      4000'
              35    xz6300'
              36    xz0016'
              37    xz0300'
,0000001'     38   wwwwwqj'
              39     s1859'  dummy instr
              40     b1918'  b(op symbol)
              41     s1810'  xz0100 Lo - 1 tr.
              42     y1943'  b[  ]
              43     bzzzz'  symbol
              44     h1821'  SYMBOL OUTPUT
              45     b1938'  -5 at 29
              46     h1903'  counter
              47     b1821'  word
              48     m1934'  1 at 17
              49     e1935'  xz6300
              50     y1954'  p[  ]
              51     e1937'  xz0300
              52     s1830'  1 at 29
              53  800t1956'  abort/non-input char
              54     pzzzz'  alpha character
,0000001'     55         2'  1 at 30 delay
              56     b1821'  word
              57     n1936'  1 at 25
              58     h1821'  word
              59     b1903'  counter
              60     a1830'  1 at 29
              61     t1946'  5 characters?
              62     b1841'  location
              63     u1801'  to incr location
.0000000'
```

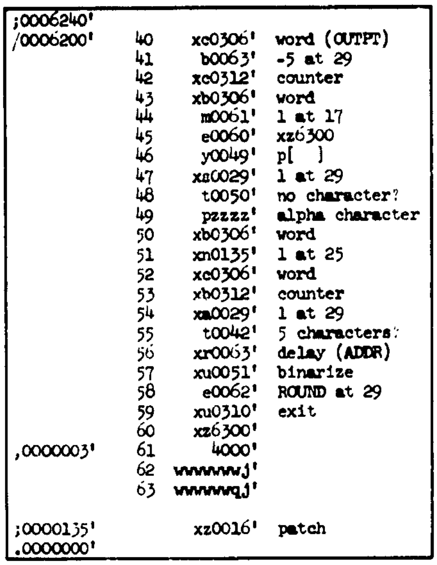
Fig. 6 - "Cast of Characters" program



```
.0004000
dimcomp800
slcr
reada
4floa;fa
tab
readb
3flob;fb
tab
fprt[sqrt[faf+fb]]
usesl

i 0322  f 0356

s01  0322
PUNCH ON:
;max4039 ,0300 less
0300   b0315'
0301   u0318'
0302 ,0000001' 00000300'
0303   u0305'
0304 ,0000001' 00000004'
0305   u0305'
0306 ,0000001' 00000000'
0307 ,0000001' 20000000'
0308 ,0000001' 00000004'
0309 ,0000001' 00000000'
0310  xu0644'
0311 ,0000001' 00000000'
0312 ,0000001' 00000000'
0313 ,0000001' 00000000'
0314 ,0000001' 00000002'
0315   a0315'
0316   h0302'
0317   u0300'
0318   a0304'
0319   y0305'
0320   b0306'
0321   u0302'
0322  xp1600'
```

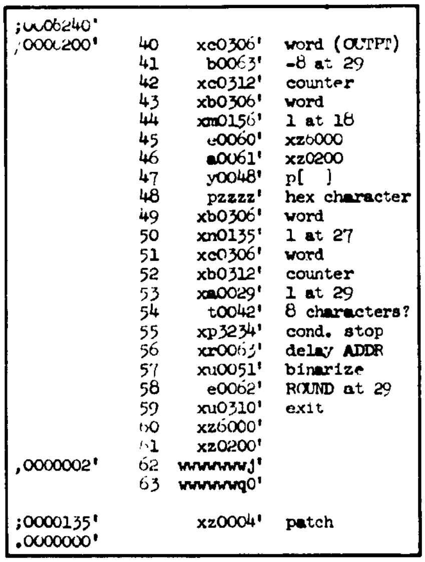
```
0323 ,0000001' 00000000'
0324    r0310'
0325   xu5900'
0326   xh6000'a
0327   xb6200'
0328    h0312'
0329   xb6000'a
0330    r0310'
0331   xu5137'
0332   xh6001'fa
0333   xp2400'
0334 ,0000001' 00000000'
0335    r0310'
0336   xu5900'
0337   xh6002'b
0338   xb6201'
0339    h0312'
0340   xb6002'b
0341    r0310'
0342   xu5137'
0343   xh6003'fb
0344   xp2400'
0345 ,0000001' 00000000'
0346   xb6001'fa
0347    h0312'
0348   xb6003'fb
0349    r0310'
0350   xu5200'
0351    r0310'
0352   xu4920'
0353    r0310'
0354   xu5700'
0355    u0322',0000001''v0053j00'
00072'01572'0000f'0150f'wwwwwwj'00002fw8'v0033q00'
00010'0000j'wwwwwwj'0000001j'v0183qf0'
01fj6qj2'wwwwwwj'q1740q30'70994'f0728'f1168'q3700q30'q2320q30'
0000j'q3kg0q30'12g44'709w0'02w80'63kj8'f1f84'13kk0'
f1f78'11814'26k7885f'26k7885f'26k7885f'26k7885f'q3q30'f3390'
26740246'
```

Fig. 7 - "Understudies" printout



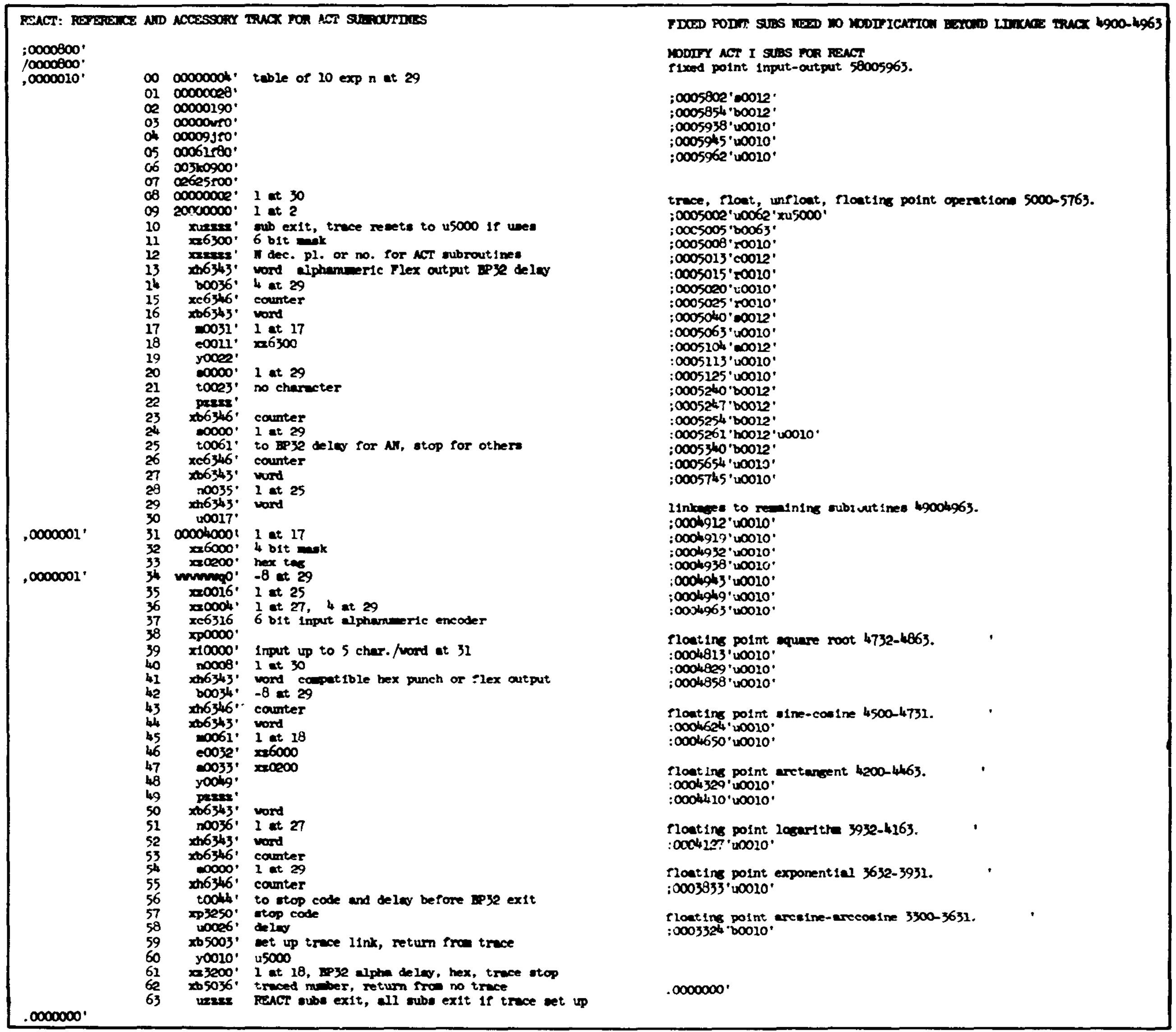
```
;0006240'
/0006200'     40    xc0306'  word (OUTPT)
              41     b0063'  -5 at 29
              42    xc0312'  counter
              43    xb0306'  word
              44     m0061'  1 at 17
              45     e0060'  xz6300
              46     y0049'  p[  ]
              47    xa0029'  1 at 29
              48     t0050'  no character?
              49     pzzzz'  alpha character
              50    xb0306'  word
              51    xn0135'  1 at 25
              52    xc0306'  word
              53    xb0312'  counter
              54    xa0029'  1 at 29
              55     t0042'  5 characters?
              56    xr0063'  delay (ADDR)
              57    xu0051'  binarize
              58     e0062'  ROUND at 29
              59    xu0310'  exit
              60    xz6300'
,0000003'     61      4000'
              62  wwwwwwj'
              63  wwwwwqj'

;0000135'           xz0016'  patch
.0000000'
```

Fig. 8 - Alphanumeric "outpt' - "addr" - "round" subroutine package



```
;0006240'
/0006200'     40    xc0306'  word (OUTPT)
              41     b0063'  -8 at 29
              42    xc0312'  counter
              43    xb0306'  word
              44    xm0156'  1 at 18
              45     e0060'  xz6000
              46     a0061'  xz0200
              47     y0048'  p[  ]
              48     pzzzz'  hex character
              49    xb0306'  word
              50    xn0135'  1 at 27
              51    xc0306'  word
              52    xb0312'  counter
              53    xa0029'  1 at 29
              54     t0042'  8 characters?
              55    xp3234'  cond. stop
              56    xr0063'  delay ADDR
              57    xu0051'  binarize
              58     e0062'  ROUND at 29
              59    xu0310'  exit
              60    xz6000'
              61    xz0200'
,0000002'     62  wwwwwwj'
              63  wwwwwq0'

;0000135'           xz0004'  patch
.0000000'
```

Fig. 9 - Compatible hex "outpt" - "addr" - "round" subroutine package



```
REACT: REFERENCE AND ACCESSORY TRACK FOR ACT SUBROUTINES

;0000800'
/0000800'
,0000010'        00  00000004'  table of 10 exp n at 29
                 01  00000028'
                 02  00000190'
                 03  00000wf0'
                 04  00009jf0'
                 05  00061f80'
                 06  003k0900'
                 07  02625f00'
                 08  00000002'  1 at 30
                 09  20000000'  1 at 2
                 10    xuzzzz'  sub exit, trace resets to u5000 if uses
                 11    xz6300'  6 bit mask
                 12    xzzzzz'  N dec. pl. or no. for ACT subroutines
                 13    xb6343'  word  alphanumeric Flex output BP32 delay
                 14     b0036'  4 at 29
                 15    xc6346'  counter
                 16    xb6343'  word
                 17     m0031'  1 at 17
                 18     e0011'  xz6300
                 19     y0022'
                 20     s0000'  1 at 29
                 21     t0023'  no character
                 22     pzzzz'
                 23    xb6346'  counter
                 24     s0000'  1 at 29
                 25     t0061'  to BP32 delay for AN, stop for others
                 26    xc6346'  counter
                 27    xb6343'  word
                 28     n0035'  1 at 25
                 29    xh6343'  word
                 30     u0017'
,0000001'        31  00004000'  1 at 17
                 32    xz6000'  4 bit mask
                 33    xz0200'  hex tag
,0000001'        34  wwwwwq0'   -8 at 29
                 35    xz0016'  1 at 25
                 36    xz0004'  1 at 27,  4 at 29
                 37    xc6316   6 bit input alphanumeric encoder
                 38    xp0000'
                 39    x10000'  input up to 5 char./word at 31
                 40     n0008'  1 at 30
                 41    xh6343'  word  compatible hex punch or flex output
                 42     b0034'  -8 at 29
                 43    xh6346'  counter
                 44    xb6343'  word
                 45     m0061'  1 at 18
                 46     e0032'  xz6000
                 47     s0033'  xz0200
                 48     y0049'
                 49     pzzzz'
                 50    xb6343'  word
                 51     n0036'  1 at 27
                 52    xh6343'  word
                 53    xb6346'  counter
                 54     s0000'  1 at 29
                 55    xh6346'  counter
                 56     t0044'  to stop code and delay before BP32 exit
                 57    xp3250'  stop code
                 58     u0026'  delay
                 59    xb5003'  set up trace link, return from trace
                 60     y0010'  u5000
                 61    xz3200'  1 at 18, BP32 alpha delay, hex, trace stop
                 62    xb5036'  traced number, return from no trace
                 63     uzzzz   REACT subs exit, all subs exit if trace set up
.0000000'
```

```
FIXED POINT SUBS NEED NO MODIFICATION BEYOND LINKAGE TRACK 4900-4963

MODIFY ACT I SUBS FOR REACT
fixed point input-output 58005963.

;0005802's0012'
;0005854'b0012'
;0005938'u0010'
;0005945'u0010'
;0005962'u0010'


trace, float, unfloat, floating point operations 5000-5763.
;0005002'u0062'xu5000'
;0005005'b0063'
;0005008'r0010'
;0005013'c0012'
;0005015'r0010'
;0005020'u0010'
;0005025'r0010'
;0005040's0012'
;0005063'u0010'
;0005104's0012'
:0005113'u0010'
;0005125'u0010'
;0005240'b0012'
;0005247'b0012'
:0005254'b0012'
:0005261'h0012'u0010'
;0005340'b0012'
;0005654'u0010'
;0005745'u0010'

linkages to remaining subroutines 49004963.
;0004912'u0010'
;0004919'u0010'
;0004932'u0010'
:0004938'u0010'
;0004943'u0010'
;0004949'u0010'
:0004963'u0010'

floating point square root 4732-4863.
:0004813'u0010'
;0004829'u0010'
;0004858'u0010'

floating point sine-cosine 4500-4731.
:0004624'u0010'
:0004650'u0010'

floating point arctangent 4200-4463.
:0004329'u0010'
:0004410'u0010'

floating point logarithm 3932-4163.
:0004127'u0010'

floating point exponential 3632-3931.
;0003833'u0010'

floating point arcsine-arccosine 3300-3631.
;0003324'b0010'



.0000000'
```

Fig. 10 - "React" program

UNCLASSIFIED

Naval Research Laboratory. Report 5784.
ENCORES FOR ACT I SYSTEM FOR LGP-30 DIGITAL COMPUTER COMPILER, by A. H. Rich. 17 pp. & figs., June 1, 1962.

The Algebraic Compiler and Translator I System for the LGP-30 digital computer has been modified and expanded for greater versatility. A location symbols table is generated at a new location, providing space for one of two decimal program and references printout routine. A 63-symbol operations table includes non-input typewriter operation symbols, breakpoint symbols, a compatible hex or alphanumeric output symbol, a binarization of address symbol, a flexible program input symbol, an integer-trim symbol, and a trace-one-intermediate-result symbol. Two alternate subroutines, located in unused space at the time of computing, utilize program input routine.

UNCLASSIFIED

1. Mathematical computer programming

I. Encores

II. Act I

III. Rich, A. H.

UNCLASSIFIED

Naval Research Laboratory. Report 5784.
ENCORES FOR ACT I SYSTEM FOR LGP-30 DIGITAL COMPUTER COMPILER, by A. H. Rich. 17 pp. & figs., June 1, 1962.

The Algebraic Compiler and Translator I System for the LGP-30 digital computer has been modified and expanded for greater versatility. A location symbols table is generated at a new location, providing space for one of two decimal program and references printout routine. A 63-symbol operations table includes non-input typewriter operation symbols, breakpoint symbols, a compatible hex or alphanumeric output symbol, a binarization of address symbol, a flexible program input symbol, an integer-trim symbol, and a trace-one-intermediate-result symbol. Two alternate subroutines, located in unused space at the time of computing, utilize program input routine.

UNCLASSIFIED

1. Mathematical computer programming

I. Encores

II. Act I

III. Rich, A. H.

UNCLASSIFIED

Naval Research Laboratory. Report 5784.
ENCORES FOR ACT I SYSTEM FOR LGP-30 DIGITAL COMPUTER COMPILER, by A. H. Rich. 17 pp. & figs., June 1, 1962.

The Algebraic Compiler and Translator I System for the LGP-30 digital computer has been modified and expanded for greater versatility. A location symbols table is generated at a new location, providing space for one of two decimal program and references printout routine. A 63-symbol operations table includes non-input typewriter operation symbols, breakpoint symbols, a compatible hex or alphanumeric output symbol, a binarization of address symbol, a flexible program input symbol, an integer-trim symbol, and a trace-one-intermediate-result symbol. Two alternate subroutines, located in unused space at the time of computing, utilize program input routine.

UNCLASSIFIED

1. Mathematical computer programming

I. Encores

II. Act I

III. Rich, A. H.

UNCLASSIFIED

Naval Research Laboratory. Report 5784.
ENCORES FOR ACT I SYSTEM FOR LGP-30 DIGITAL COMPUTER COMPILER, by A. H. Rich. 17 pp. & figs., June 1, 1962.

The Algebraic Compiler and Translator I System for the LGP-30 digital computer has been modified and expanded for greater versatility. A location symbols table is generated at a new location, providing space for one of two decimal program and references printout routine. A 63-symbol operations table includes non-input typewriter operation symbols, breakpoint symbols, a compatible hex or alphanumeric output symbol, a binarization of address symbol, a flexible program input symbol, an integer-trim symbol, and a trace-one-intermediate-result symbol. Two alternate subroutines, located in unused space at the time of computing, utilize program input routine.

UNCLASSIFIED

1. Mathematical computer programming

I. Encores

II. Act I

III. Rich, A. H.